Assignment 4

Yifei Gu

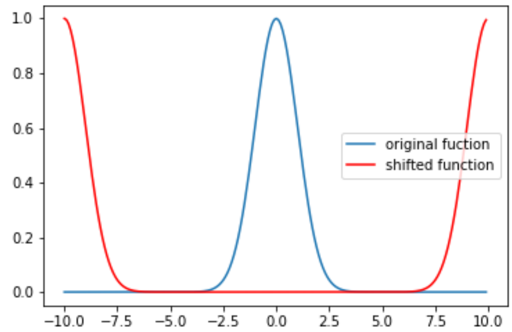
260906596

Q1)

Here is my defined function and choice of gaussian function:

```
def shift(y,dx):
    N=y.size
    kvec=np.arange(N)
    yft=np.fft.fft(y)
    J=np.complex(0,1)
    yft_shift=yft*np.exp((-2*np.pi*J*kvec*dx)/N)
    y_shift=np.fft.ifft(yft_shift)
    return y_shift

x=np.arange(-10,10,0.1)
y=np.exp(-0.5*x**2)
dx=100.0
y_shift=shift(y,dx)
```



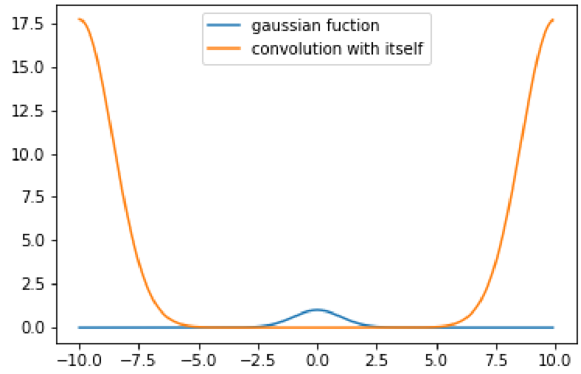
The gaussian is shifted by half of the array length.

Q2)

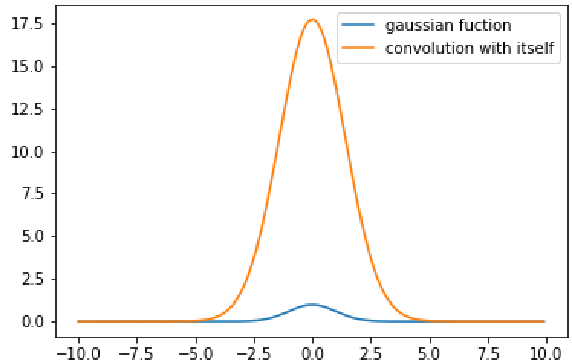
```
def conv(f,g):
    ft_f=np.fft.fft(f)
    ft_g=np.fft.fft(g)
    ft_g=np.conj(ft_g)
    convolution_before_shift=np.fft.ifft(ft_f*ft_g)
    convolution=np.fft.fftshift(convolution_before_shift)
    return convolution, convolution_before_shift
```

The result from **np.fft.fft** is not centered. Considering the periodicity, I used

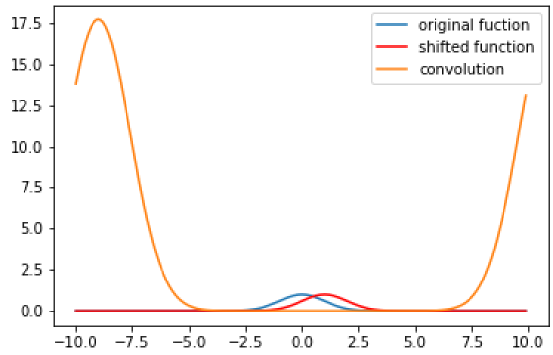
np.fft.fftshift to **move** the convolution **to the center**. Here I showed both plots.



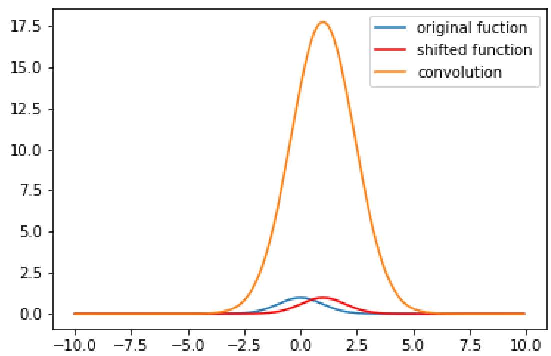
The gaussian I chose is **y= exp(-0.5*x**2)**. This plot is not easy to interpret. However, when shift the convolution, which is a gaussian, to the center, it makes much more sense now. A **fat gaussian** becomes a **skinny gaussian** after Fourier Transform.



Q3)



Again, I used **np.fft.fftshift** to move the convolution to the center. Comparing with the convolution curve with guassian itself, the convolution curve between gaussian and its shifted gaussian **moves** towards the shifted gaussian.

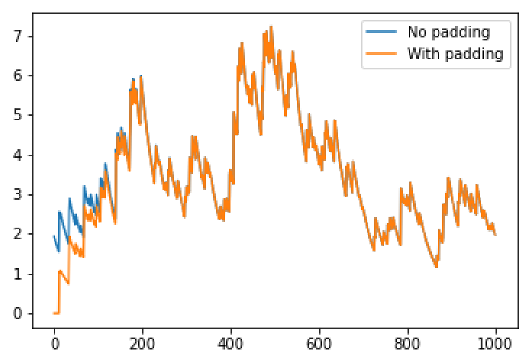


Q4)

```
# NO padding, wrapi-around with danger
f1=0.0*x
for i in range(nhit):
    f1[x_hit[i]]=f1[x_hit[i]]+y_hit[i]
g1=np.exp(-1.0*x/tau)
T1=np.fft.irfft(np.fft.rfft(g1)*np.fft.rfft(f1))
T1=T1[:len(x)]
```

Here I used the example showed in class. Doing convolution with no padding might cause the problem of wrap-around, which is due to the periodicity of Fourier transfer. As you can see from the plot, the curve without padding is trying to connect the starting point and ending point. **Padding the data with some zeros and chopping them off in the end will solve this problem.**

```
#Padding, wrap-around without danger
f2=0.0*x
f2=np.pad(f2,[0,len(f2)])
for i in range(nhit):
    f2[x_hit[i]]=f2[x_hit[i]]+y_hit[i]
g2=np.exp(-1.0*x/tau)
g2=np.pad(g2,[0,len(g2)])
T2=np.fft.irfft(np.fft.rfft(g2)*np.fft.rfft(f2))
T2=T2[:len(x)]
```



Q5)

a)

Let $\alpha = exp(-2\pi ik/N)$,

$$\sum_{x=0}^{N-1} exp(-2\pi ikx/N) = \sum_{x=0}^{N-1} \alpha^x = S = 1 + \alpha + \alpha^2 + \cdots + \alpha^{N-1}$$

$$S * \alpha = \alpha + \alpha^2 + \cdots + \alpha^N$$
$$S - S * \alpha = 1 - \alpha^N$$
$$S * (1 - \alpha) = 1 - \alpha^N$$
$$S = \frac{1 - \alpha^N}{(1 - \alpha)}$$

Bring $\alpha = exp(-2\pi ik/N)$ back to the equation:

$$\sum_{x=0}^{N-1} exp(-2\pi ikx/N) = \frac{1 - exp(-2\pi ik/N)^N}{1 - exp(-2\pi ik/N)}$$
$$= \frac{1 - exp(-2\pi ik)}{1 - exp(-2\pi ik/N)}$$

b)

According to *L'Hôpital*'s rule, when k→0:

$$\lim_{k\to 0} \sum_{x=0}^{N-1} exp(-2\pi ikx/N) = \lim_{k\to 0} \frac{1 - exp(-2\pi ik)}{1 - exp(-2\pi ik/N)}$$
$$= \lim_{k\to 0} \frac{(1 - exp(-2\pi ik))'}{1 - exp(-2\pi ik/N)'}$$
$$= \lim_{k\to 0} \frac{2\pi i * exp(-2\pi ik)}{(2\pi i/N) * exp(-2\pi ik/N)}$$
$$= \lim_{k\to 0} \frac{exp(-2\pi ik)}{\exp\left(-\frac{2\pi ik}{N}\right)} N$$
$$= \frac{exp(0)}{\exp(0)} N$$
$$= N$$

When integer k is not a multiple of N, k/N will not be an integer. $exp(-2\pi ik) = 1$,

but $\exp\left(-\frac{2\pi ik}{N}\right) \neq 1$. $\frac{1 - exp(-2\pi ik) \to 0}{1 - exp\left(-\frac{2\pi ik}{N}\right) \to \neq 0}$ = 0.

c)

According to **Euler's formula**:

$$F(k') = \sum_{x=0}^{N-1} \sin\left(\frac{2\pi ikx}{N}\right) * \exp\left(-2\pi ik'x/N\right)$$

$$= \sum_{x=0}^{N-1} \frac{\exp\left(\frac{-2\pi ikx}{N}\right) - \exp\left(\frac{2\pi ikx}{N}\right)}{2i} * \exp\left(-2\pi ik'x/N\right)$$

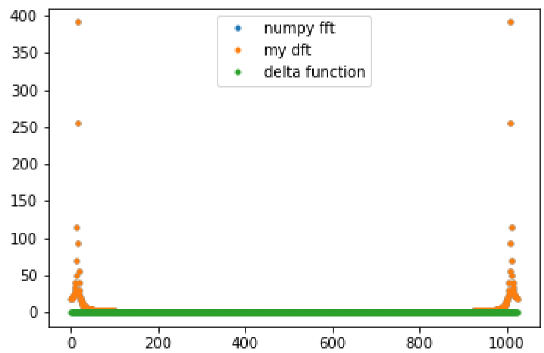
$$= \sum_{x=0}^{N-1} \frac{\exp\left(\frac{-2\pi i(k'+k)x}{N}\right)}{2i} - \frac{\exp\left(\frac{-2\pi i(k'-k)x}{N}\right)}{2i}$$

With this **analytic estimate of the DFT**, I wrote my DFT function as below:

```
def mydft(k_sin,y):
    N = len(y)
    x=np.arange(N)
    kvec = np.fft.fftfreq(N,1/N)
    F = np.zeros(N)
    for i in range(len(kvec)):
        F_i= np.sum(np.exp(-2*np.pi*1j*(kvec[i]-k_sin)*x/N)/2j-np.exp(-2*np.pi*1j*(kvec[i]+k_sin)*x/N)/2j)
        F[i] = abs(F_i)
    return F, kvec
```

I chose **a non-integer k =15.4**, which will reflect as **a peak** in the Fourier Transform of the sine wave near **k'=15**. Due to aliasing, there be **another peak near k'=1009**, which is the **mirror image** about **N/2** of **k'=15**. **This is also consistent with the analytic estimate of DFT.** When **k'=k** or **k'=-k**, $\exp\left(\frac{-2\pi i(k'+k)x}{N}\right)$ and $\exp\left(\frac{-2\pi i(k'-k)x}{N}\right)$ are not zero and **all the other k' will have a DFT equal to zero.** Here is how I defined my **delta function**:

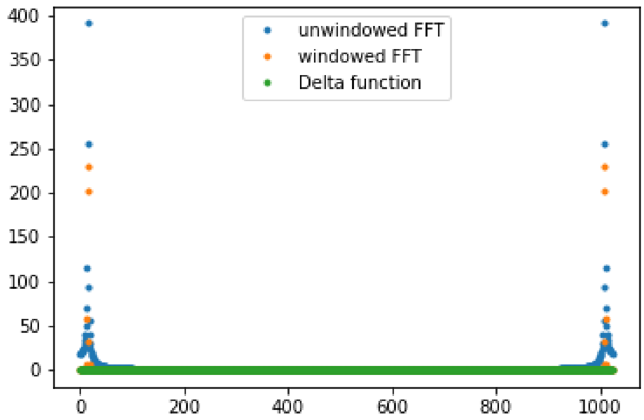
```
delta = np.zeros(len(yft2))
delta[15]=512 #N/2=512
delta[1011]=512 #N/2=512
```



As you can see, the **numpy fft**(blue) is **overlapping** with "**my dft**", but both have some **spectral leakage** from the delta function. The error **between my written DFT and numpy FFT** is close to the **machine precision**, and the error **between my written DFT and delta** function is **huge**:

```
Error between my written DFT and numpy FFT= 9.403194348437380e-13
Error between my written DFT and delta function= 23.63514640992592
```

d)



With windowing (Orange dots), the **spectral leakage** effects are **reduced**. The **error** between FFT and delta function also **dropped**.

```
Error between delta and FFT without window 23.635146409925916
Error between delta and FFT with window 21.599791805286124
```

e)

$$F(k') = \sum_{x=0}^{N-1} (0.5 - 0.5\cos(2\pi x/N)) * \exp\left(-2\pi i k' x/N\right)$$

$$F(k') = \sum_{x=0}^{N-1} (0.5 - 0.5\frac{\exp\left(\frac{2\pi i x}{N}\right) + \exp\left(-\frac{2\pi i x}{N}\right)}{2}) * \exp\left(-2\pi i k' x/N\right)$$

$$F(k') = \sum_{x=0}^{N-1} 0.5\exp\left(-\frac{2\pi i k' x}{N}\right) - 0.5\frac{\exp\left(\frac{2\pi i x(k'-1)}{N}\right)}{2} - 0.5\frac{\exp\left(-\frac{2\pi i x(k'+1)}{N}\right)}{2}$$

When k'=0, F(0)=N/2,
When k'=1, F(0)=-N/4,
When k'=-1, F(0)=-N/4,
All the other k' will give a zero for DFT. The Fourier Transform of the window is [N/2, -

N/4,0,0······0,-N/4].

$$F(f * window) = F(f)F(window)$$